**Управление качеством программных систем. Донскова Грета, ПИ21-3**

**ЛАБОРАТОРНАЯ РАБОТА №2 (8).**

Тема лабораторной работы: работа с классификацией видов тестирования.

**Документация к веб-приложению с курсами повышения квалификации**

**Цель работы:**

Целью данной работы является проведение анализа ранее разработанного проекта системы управления курсами обучения. В рамках анализа будут рассмотрены различные уровни, типы, методы и виды тестирования, применимые к данному проекту, а также приведены примеры характерных дефектов и сделаны выводы по результатам работы.

**Описание программного проекта:**

Программный проект представляет собой систему управления курсами обучения, реализованную как клиент-серверное приложение:

* **Серверная часть:** Реализована с использованием Spring Boot. Взаимодействие с базой данных MySQL осуществляется через ORM.
* **Клиентская часть:** Реализована с использованием HTML и интегрированных библиотек и фреймворков для создания удобного интерфейса пользователя.
* **Основные функции:** Отображение данных о курсах в таблице, заполнение таблицы корректными данными, фильтрация и сортировка записей, загрузка данных из базы данных при запуске программы.
* **Дополнительные возможности:** Анализ статистических данных через отображение гистограммы и таблицы с информацией о количестве курсов в месяц.

**По каждому из элементов классификации, описанному в разделе методических рекомендаций, привести список видов, типов, методов, уровней тестирования, применимых для выбранного проекта:**

**Уровни тестирования:**

1. **Компонентное (модульное) тестирование:**
   * **Цель:** Выявление локализованных в модуле ошибок и определение степени готовности системы к интеграции.
   * **Методы:** JUnit тесты для Java, тестирование отдельных классов и методов.
   * **Примеры дефектов:** Ошибки в алгоритмах сортировки и фильтрации, неправильное отображение данных в таблице.
2. **Интеграционное тестирование:**
   * **Компонентный интеграционный уровень:**
     + **Цель:** Проверка взаимодействия между компонентами системы.
     + **Методы:** Spring Test для тестирования взаимодействия между слоями (контроллеры, сервисы, репозитории).
     + **Примеры дефектов:** Неправильное связывание компонентов, ошибки в передачах данных между слоями.
   * **Системный интеграционный уровень:**
     + **Цель:** Проверка взаимодействия с внешними системами и компонентами.
     + **Методы:** MockMvc для тестирования REST API, тестирование взаимодействия с базой данных.
     + **Примеры дефектов:** Ошибки при взаимодействии с MySQL, некорректные ответы API.
3. **Системное тестирование:**
   * **Цель:** Проверка функциональных и нефункциональных требований к системе в целом.
   * **Методы:** End-to-End тестирование с использованием Selenium или Cypress.
   * **Примеры дефектов:** Непредусмотренные сценарии использования, неудобство интерфейса, неправильная работа под нагрузкой.

**Типы тестирования:**

1. **Функциональное тестирование:**
   * **Методы:** Тестирование пользовательских сценариев, проверка корректности работы функций (создание, обновление, удаление курсов).
   * **Примеры дефектов:** Некорректная обработка пользовательских данных, ошибки в логике бизнес-процессов.
2. **Нефункциональное тестирование:**
   * **Нагрузочное тестирование:** Проверка системы под максимальной нагрузкой.
     + **Примеры дефектов:** Падение производительности, долгий отклик системы.
   * **Стресс-тестирование:** Проверка системы в экстремальных условиях.
     + **Примеры дефектов:** Сбой системы при резком увеличении количества запросов.
   * **Тестирование удобства пользования:** Оценка UI/UX.
     + **Примеры дефектов:** Сложность навигации, неудобное расположение элементов интерфейса.

**Методы тестирования:**

1. **Статические методы:**
   * **Рецензирование кода:** Анализ кода вручную и с помощью инструментов (например, SonarQube).
   * **Примеры дефектов:** Пропущенные требования, стилистические ошибки в коде.
   * **Статический анализ:** Проверка кода на возможные уязвимости и ошибки.
     + **Примеры дефектов:** Неиспользуемый код, возможные утечки памяти.
2. **Динамические методы:**
   * **Методы белого ящика:** Тестирование с полным знанием внутренней структуры системы.
     + **Примеры дефектов:** Локализация ошибок, связанных с внутренними зависимостями.
   * **Методы черного ящика:** Тестирование без знания внутренней структуры системы.
     + **Примеры дефектов:** Неверная обработка входных данных.
   * **Методы серого ящика:** Комбинирование белого и черного ящика.
     + **Примеры дефектов:** Ошибки, возникающие при переходе от одной функциональности к другой.

**Привести примеры дефектов, характерных для каждого вида тестирования:**

* **Компонентное тестирование:**
  + При добавлении нового курса с пустым названием метод не возвращает ошибку, что приводит к некорректным записям в базе данных.
  + Метод вычисления среднего рейтинга курса не обрабатывает корректно пустой список отзывов, что вызывает ошибку выполнения.
* **Интеграционное тестирование:**
  + Некорректная обработка исключений в случае недоступности базы данных приводит к падению приложения и потере данных.
  + Взаимодействие между контроллером и сервисом приводит к циклическим зависимостям и ошибке при загрузке контекста Spring.
* **Системное тестирование:**
  + Интерфейс не адаптируется под мобильные устройства, что делает систему неудобной для пользователей смартфонов.
  + При загрузке большого объема данных из базы происходит утечка памяти, что приводит к краху приложения и потере данных.
* **Нагрузочное тестирование:**
  + Увеличение времени отклика сервера при росте числа одновременных запросов до 2 секунд, что снижает производительность и удобство использования.
* **Стресс-тестирование:**
  + Потеря данных при одновременном выполнении большого количества транзакций, что приводит к несоответствию данных в системе.
* **Тестирование удобства пользования:**
  + Кнопка "Сохранить" не видна на некоторых экранах из-за плохой адаптации интерфейса, что затрудняет сохранение данных пользователем.
  + Поля ввода данных не имеют подсказок, что затрудняет их заполнение и приводит к ошибкам ввода.

**Выводы по работе**

Проанализированный проект системы управления курсами обучения демонстрирует комплексный подход к реализации программного обеспечения с использованием современных технологий и инструментов. Проведенный анализ показал необходимость применения различных уровней, типов и методов тестирования для обеспечения высокого качества и надежности системы. Основное внимание следует уделить как функциональным, так и нефункциональным аспектам тестирования, что позволит своевременно выявлять и устранять дефекты на ранних этапах разработки.

**Список использованных источников**

1. Руководство по использованию MockMvc (Дата обращения: 17.05.2024)

https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/test/web/servlet/MockMvc.html

1. Статья “ Наглядное руководство по каждому типу тестов” (Дата обращения: 17.05.2024)

https://medium.com/nuances-of-programming/наглядное-руководство-по-каждому-типу-тестов-которые-вы-можете-написать-180c63c0b94f

1. Статья “ Фундаментальная теория тестирования” (Дата обращения: 17.05.2024)

https://habr.com/ru/articles/549054/